PRE-APPEAL BRIEF REQUEST FOR REVIEW	**Docket Number:** P19009	
I hereby certify that this correspondence is being transmitted via the EFS-Web System to the USPTO on: April 13, 2009	**Application Number:** 10/823,895	**Filed:** April 13, 2004
Signature: /David Victor/	**First Named Inventor:** M.A. ROTHMAN et al.	
Typed or Printed Name: David W. Victor	**Art Unit:** 2185	**Examiner:** Jae Un Yu

Applicant requests review of the final rejection in the above-identified application. No amendments are being filed with this request.

This request is being filed with a notice of appeal.

The review is requested for the reason(s) stated on the attached five (5) sheet(s).
Note: No more than five (5) pages may be provided.

I am the:

☐ applicant/inventor	/David Victor/ Signature
☐ assignee of record of the entire interest. See 37 CFR 3.71. Statement under 37 CFR 3.73(b) is enclosed. (Form PTO/SB/96)	David W. Victor Typed or Printed Name
☒ attorney or agent of record. Registration Number Registration No. 39,867	(310) 553-7977 Telephone Number
☐ attorney or agent acting under 37 CFR 1.34 Registration number if acting under 37 CFR 1.34 ________	April 13, 2009 Date

NOTE: Signatures of all the inventors or assignees of record of the entire interest or their representative(s) are required. Submit multiple forms if more than one signature is required*.

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant(s):	M.A. ROTHMAN et al.	Examiner	Jae Un Yu
Serial No.	10/823,895	Group Art Unit	2185
Filed	April 13, 2004	Docket No.	P19009
TITLE	DEFRAGMENTING OBJECTS IN A STORAGE MEDIUM		

PRE-APPEAL BRIEF REQUEST FOR REVIEW ARGUMENTS

Applicants request review of Examiner's rejection of the claims as obvious (35 U.S.C. §103) over Lawrence (U.S. Patent No. 6,253,300) in view of combinations with Andrew (U.S. Patent App. No. 2004/0059863), Brown (U.S. Patent No. 6,038,636), Douglis (U.S. Patent Pub. No. 2005/018075), and Ball (U.S. Patent Pub. No. 2005/0162944) in the Final Office Action dated December 12, 2008 ("FOA").

With respect to claim 1, 10, 19, and 23, Applicants request review of the Examiner's finding that col. 5, lines 37-42 of Lawrence teaches the claim requirement of defragmenting the object in storage so that blocks in storage including the object are contiguous in response to receiving the I/O request. (FOA5, pgs. 2-3)

The cited col. 5 of Lawrence mentions that each file is stored in several locations separated by regions of the storage medium that do not hold the file's contents and that fragmentation can be alleviated or eliminated by running a defragmentation program on the files before copying them. Nowhere does this cited col. 5 teach the claim requirement that an I/O request to write an update to the object causes defragmentation of the object. Instead, the cited col. 5 mentions that a defragmentation program can be run on files before copying them. Although one may run a defragmentation program at any time, after or before copying data, the cited col. 5 still does not teach defragmenting an object in response to receiving an I/O request to write the update the object to which the defragmentation is directed.

The Examiner cited FIG. 4 of Andrew as triggering a defragmentation in response to an I/O request because a shutdown operation inherently comprises disk write operations in that the shutdown operation will save/write system state information. (FOA, pg. 3 and 11) The cited FIG. 4 mentions that a hard drive is defragmented as part of a shutdown operation after temporary Internet files are removed. (Andrews, para. 35). However, the claims require that an object subject to an I/O operation is defragmented in response to an I/O request to update that object. The cited Andrews discusses general defragmenting as part of a shut down operation.

However, nowhere does the cited Andrews teach or mention that an object is defragmented in response to an update to that object. In other words, it is not the write to the system state information that triggers the defragmentation of the system as a whole in Andrews as the Examiner suggests, but a shutdown operation. Also, there is no teaching in the cited Andrews that a specific file is defragmented in response to an update to that file. Instead, the cited Andrews discusses defragmenting the entire hard drive as part of a shutdown, not defragment a particular object in response to a write to that object.

With respect to claims 2, 11, and 24, Applicants request review of the Examiner's finding that col. 5, lines 37-39 of Lawrence teaches that the I/O request is executed with respect to the object after defragmenting the object. (FOA, pg. 4) The cited col. 5 mentions defragmenting files before copying them. This does not teach or suggest updating the object after defragmenting the object. The Examiner found that the distinction of Lawrence and the claim requirement of executing the I/O request to the object after defragment is patentably indistinct and that Andrews teaches executing the write after defragments. (FOA, pg. 12) Applicants submit that defragmenting an object after updating the object is different and not taught by defragmenting a file before a write. Also, Applicants note that the cited FIG. 4 of Andrews does not teach when the "writing" occurs, before or after defragmentation, because the Examiner found this writing step "inherent", not expressly taught.

With respect to dependent claims 8, 17, and 30, Applicants request review of the Examiner finding that Lawrence inherently discloses the claimed operations of receiving the I/O request, initiating the operation to defragment the object, and executing the I/O request of defragmenting the object in storage are performed by a storage controller managing I/O requests to the storage. (FOA, pgs. 4-5)

Applicants submit that there is nothing inherent that defragmentation be initiated by the storage controller as opposed to some other computer component. According to the Manual of Patent Examination and Procedure (MPEP), the "fact that a certain result or characteristic may occur or be present in the prior art is not sufficient to establish the inherency of that result or characteristic." MPEP Sec. 2112, pg. 57 (Aug. 2005, Rev. 3). Thus, the fact that defragmentation "may" be initiated in the storage controller as opposed to a program in the computer makes this finding of inherency inappropriate. Applicants submit that although computers may have a storage controller as the Examiner notes, the Examiner has not cited any art that suggests that performing defragmenting of an object in response to a write to the object is performed by the storage controller. The Examiner is using hindsight to propose a modification to known computer components, such as a storage controller, that is not taught or suggested in the cited art.

With respect to dependent claims 9, 18, and 31, Applicants request review of the finding that Lawrence inherently teaches that that the operation of defragmenting the object in storage is performed by a device driver for the storage providing an interface to the storage. (FOA, p. 4-5) As with claims 8, 17, and 30, Applicants submit that the claims are patentably distinct because the Examiner has not shown where the cited Lawrence teaches that defragmentation is performed by a device driver for the storage providing an interface to the storage as opposed to some other software program, such as an application program or utility. Thus, it is not inherent that a device driver perform the defragmentation.

With respect to claim 3, 12, 20, and 25, Applicants request review of the Examiner's interpretation of the claimed threshold as zero and amount of fragmentation with respect to the threshold as exceeding zero. (FOA, pg. 12) Such an interpretation would render meaningless the claim limitation of determining whether an amount of fragmentation of the object in the storage exceeds a fragmentation threshold indicating an acceptable number of bytes stored in non-contiguous locations in response to receiving the I/O request, wherein the object is defragmented if the amount of fragmentation exceeds the fragmentation threshold, and wherein the I/O request to update the object is executed without defragmenting the object in response to determining that the amount of fragmentation does not exceed the fragmentation threshold. Thus, the recitation of a threshold requires some value other than zero and an amount of fragmentation exceeding such threshold, else the limitation has no meaning.

Applicants further request review of the finding that col. 5, lines 37-39 of Lawrence and col. 10, lines 1-5 and col. 7, lines 45-46 teaches these claim requirements. (FOA, pgs. 5-6) The cited col. 5 mentions that fragmentation can be eliminated or alleviated by running a defragmentation program on the files before copying them. This cited col. 5 does not teach determining whether an amount of fragmentation of an object exceeds a threshold indicating an acceptable number of bytes stored in non-contiguous locations in response to receiving a request to write update an object. The cited col. 7 of Brown mentions that a file header includes a number indicating that the memory is valid, the name of a file, and a pointer to the next file, a number indicating the size of the file. The cited col. 10 mentions how to determine whether a file is contiguous by determining whether the size field in the header equals a predetermined code. Although the cited Brown mentions a number indicating a size of a file and using the size field in the header to determine whether the file is contiguous, this does not teach the claim requirement of determining whether an amount of fragmentation of an object exceeds a threshold indicating an acceptable number of bytes stored in non-contiguous locations in response to receiving a request to write update an object.

With respect to dependent claims 4, 13, and 26, Applicants request review of the Examiner finding that para. [0032] of Douglis teaches the claim requirement determining whether a user settable flag indicates to perform defragmentation in response to receiving the I/O request, wherein the object is defragmented if the flag indicates to perform defragmentation. (FOA, pgs. 9-10, 13). The cited para. [0032] discusses a power-aware monitor that monitors applications to defer execution of non-critical background tasks, that may be daemons or other application and whose execution is desirable only when there is not a restriction on power usage. Examples include full disk virus scans and defragmentation, among others. Although the cited para. [0032] discusses a power monitor deferring defragmentation to execute when there is no restriction on power usage, the cited para. [0032] does not teach or suggest a user settable flag that indicates to perform defragmentation in response to receiving the I/O request, which is to update the object. Instead, the cited para. [0032] discusses deferring defragmentation for power management concerns.

With respect to dependent claims 6, 15, and 28, Applicants request review of the Examiner finding that the Abstract, the object 24, and para. 24 teaches the claim requirement of determining at least one logical partition including the object, wherein the object is defragmented if the object is within one logical partition and the I/O request to update the object is executed without defragmenting the object in response to determining that the object is included in more than one logical partition. (FOA, pgs. 9-10) The cited Abstract discuses a redundant memory architecture having an active memory and an inactive memory. The active memory supports in-service storage operations. The inactive memory is updated with stored contents of the active memory. Stored contents of the inactive memory are defragmented prior to an activity switch that results thenceforth in the inactive memory assuming the in-service storage operations and the active memory being updated with the stored contents of the inactive memory. The cited para. [0024] of Ball mentions that the defragmentation can be performed on an inactive redundant memory, such that the in-service performance of a counterpart active memory need not be impacted.

The cited Ball does not teach or suggest defragmenting the object to update in response to determining that the object is included within one logical partition. Instead, the cited Abstract mentions that the inactive memory is defragmented prior to an activity switch that results in the inactive memory assuming the in-service storage operations and that the defragmentation can be performed on an inactive redundant memory.

In the Response, the Examiner found that Andrews teaches defragment within a disk, which corresponds to one logical portion. (FOA, pg. 14) However, the claims require defragment upon determining that the object subject to the update is in one logical partition, and not defragmenting if the object is included in multiple logical partitions. Although Andrews may defragment within a disk as the Examiner asserts, defragmenting in a disk does not teach or suggest deciding whether to perform defragmentation of an object based on whether the object to defragment is within one or more logical partitions.

Dated: April 13, 2009

By: /David Victor/
David W. Victor
Registration No. 39,867

Please direct all correspondences to:

David W. Victor
Konrad Raynes & Victor, LLP
315 South Beverly Drive, Ste. 210
Beverly Hills, CA 90212
Tel: (310) 553-7977
Fax: 310-556-7984